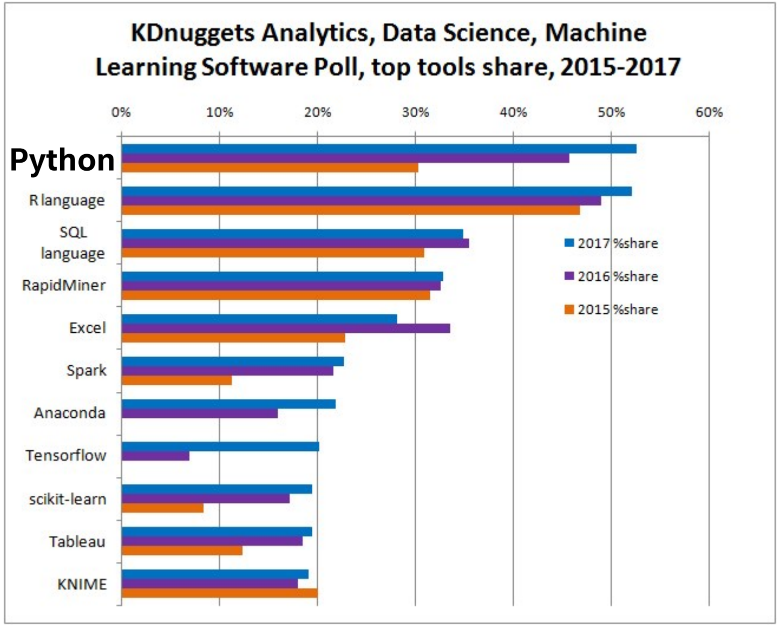
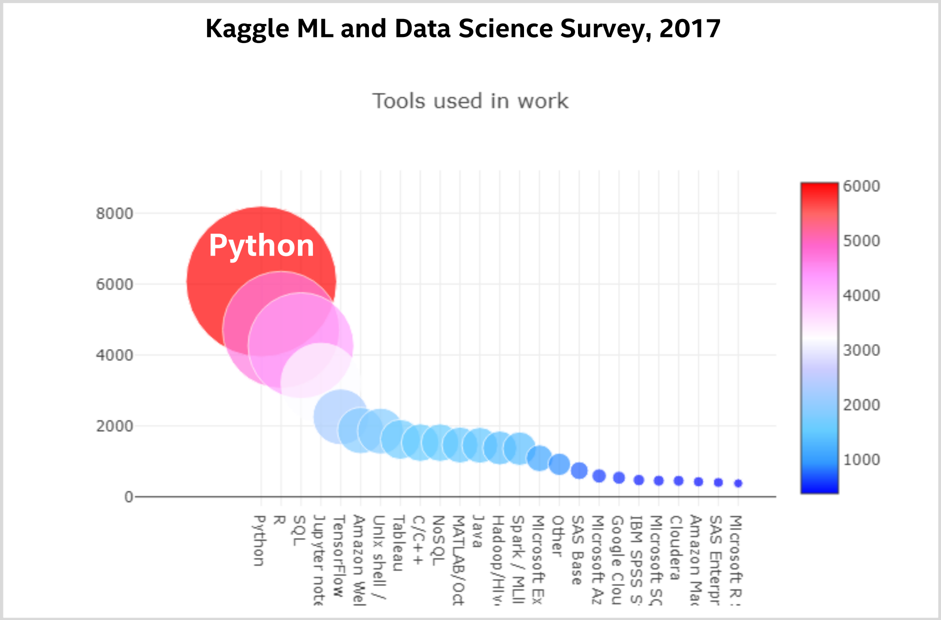
PYTHON: PRODUCTIVITY WITH PERFORMANCE

Heidi Pan

Scripting Analyzers and Tools Group (Python, R, Julia, Go)
Intel

Python for Data Science & Machine Learning





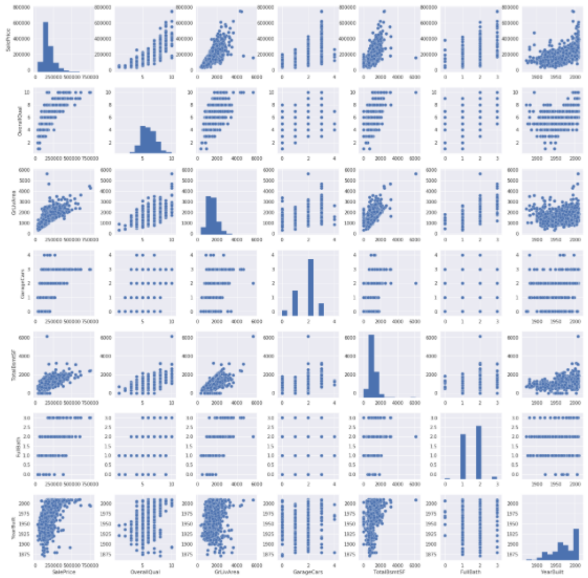
https://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning-software-leaders.html

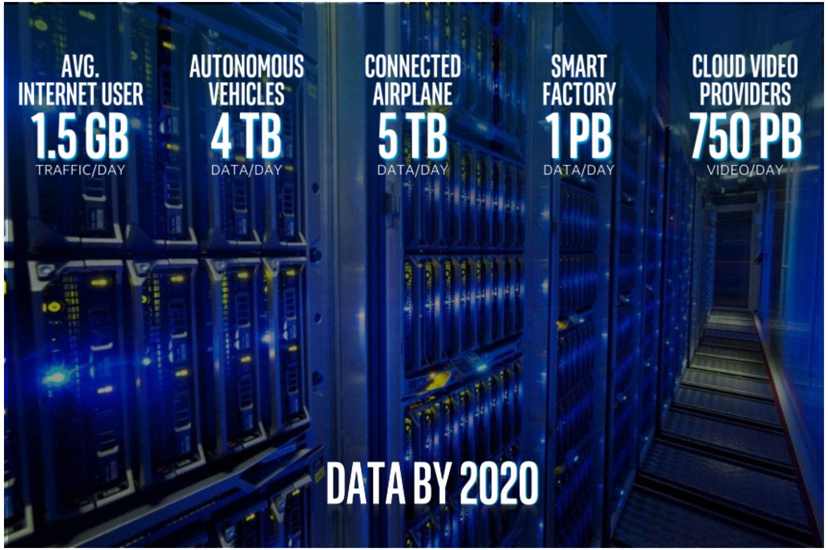
https://www.kaggle.com/sudalairajkumar/an-interactive-deep-dive-into-survey-results/data

From Prototype to Production

In [13]:

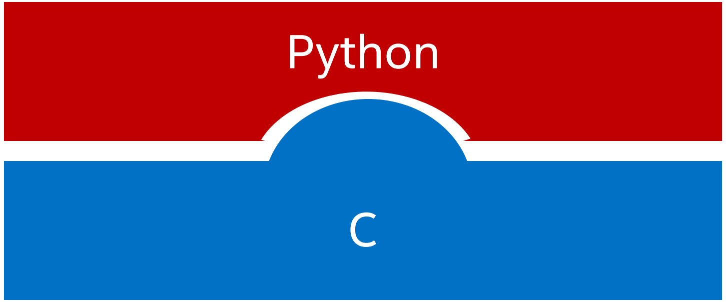
```
#scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.pairplot(df_train[cols], size = 2.5)
plt.show();
```



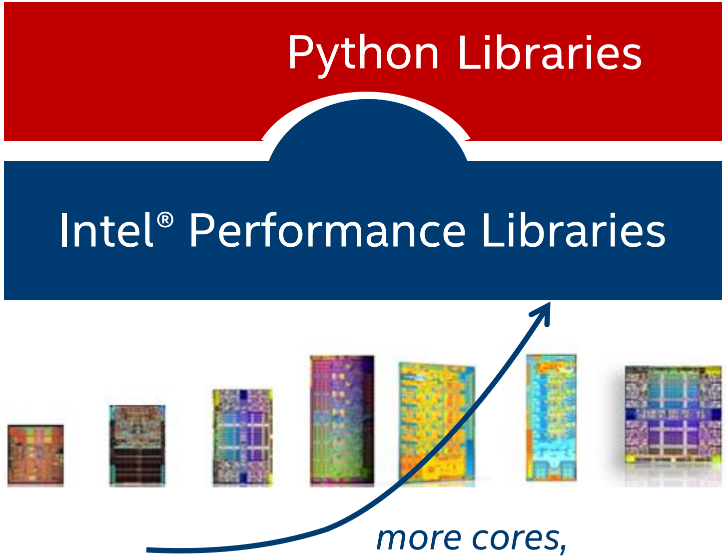


https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

High Performance Python



High Performance Python



(generations of processors)

more cores,
more threads,
wider vectors, ...

Accelerating Machine Learning

scikit-learn

Intel® Data Analytics Acceleration Library (DAAL)

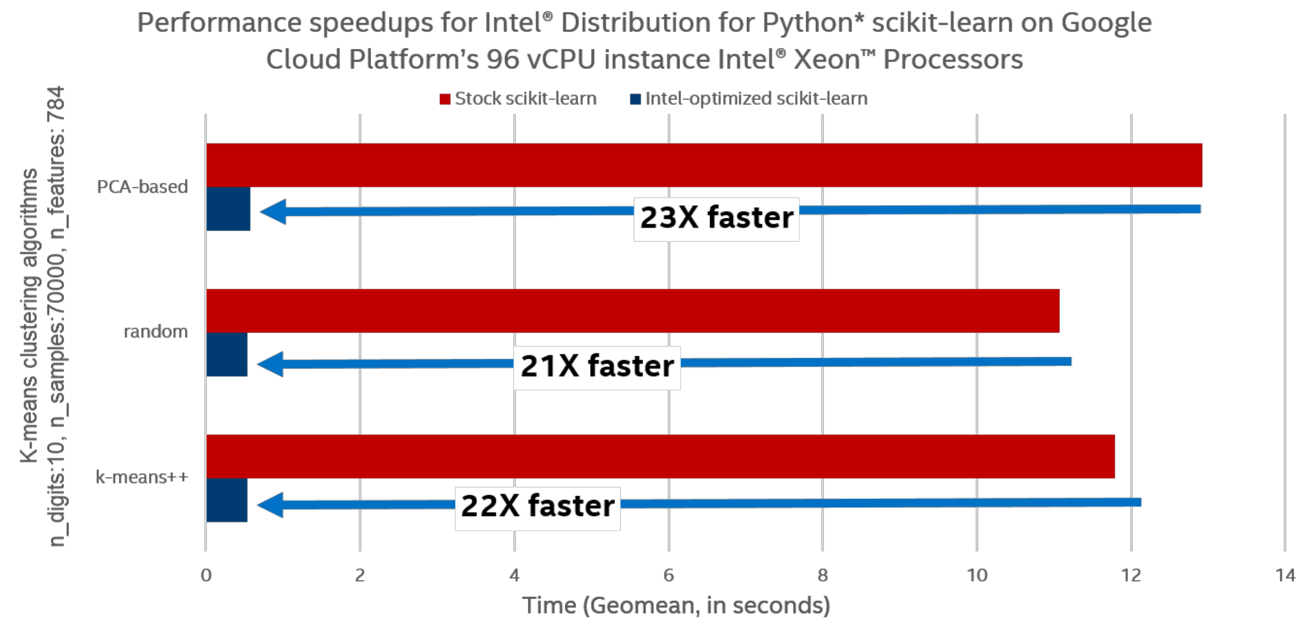
Intel® Math Kernel Library (MKL)

Intel® Threading Building Blocks (TBB)

- **Efficient memory layout** via `Numeric Tables`
- **Blocking** for optimal cache performance
- **Computation mapped to most efficient matrix operations** (in MKL)
- **Parallelization** via TBB
- **Vectorization**

Try it out! `conda install -c intel scikit-learn`

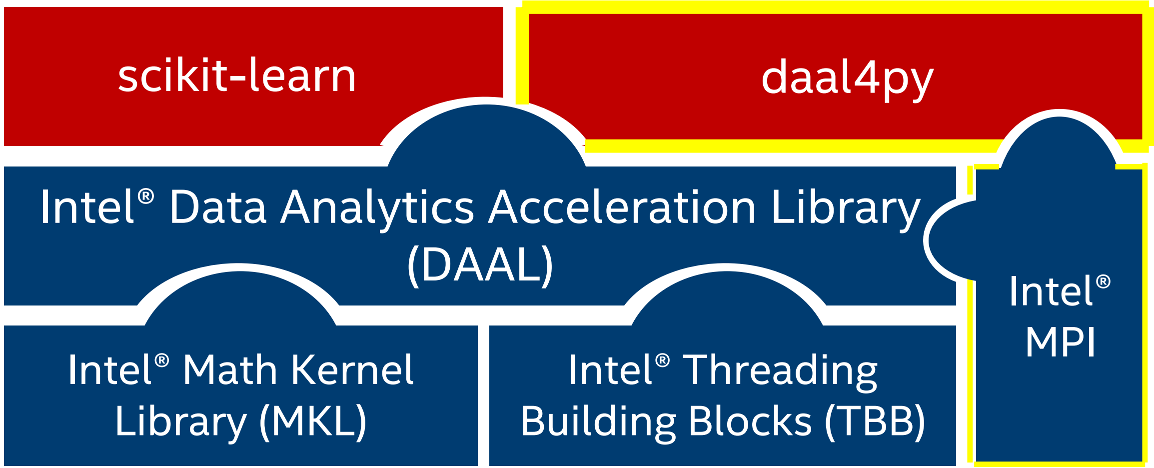
Accelerating K-Means



System Configuration: GCP VM, zone us-central1-c; 96 vCPU, Intel Skylake; 360 GB memory. Ubuntu 16.04.3 LTS; Linux instance-1 4.10.0-38-generic #42~16.04.1-Ubuntu SMP Tue Oct 10 16:32:20 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux; Intel® Distribution for Python* from Docker image intelpython/intelpython3_full:latest (created 2017-09-12T20:10:42.862965559Z); Stock Python*: pip install scikit-learn

https://cloudplatform.googleblog.com/2017/11/Intel-performance-libraries-and-python-distribution-enhance-performance-and-scaling-of-Intel-Xeon-Scalable-processors-on-GCP.html

Scaling Machine Learning Beyond a Single Node



Simple Python API

Powered by DAAL

Scalable to multiple nodes

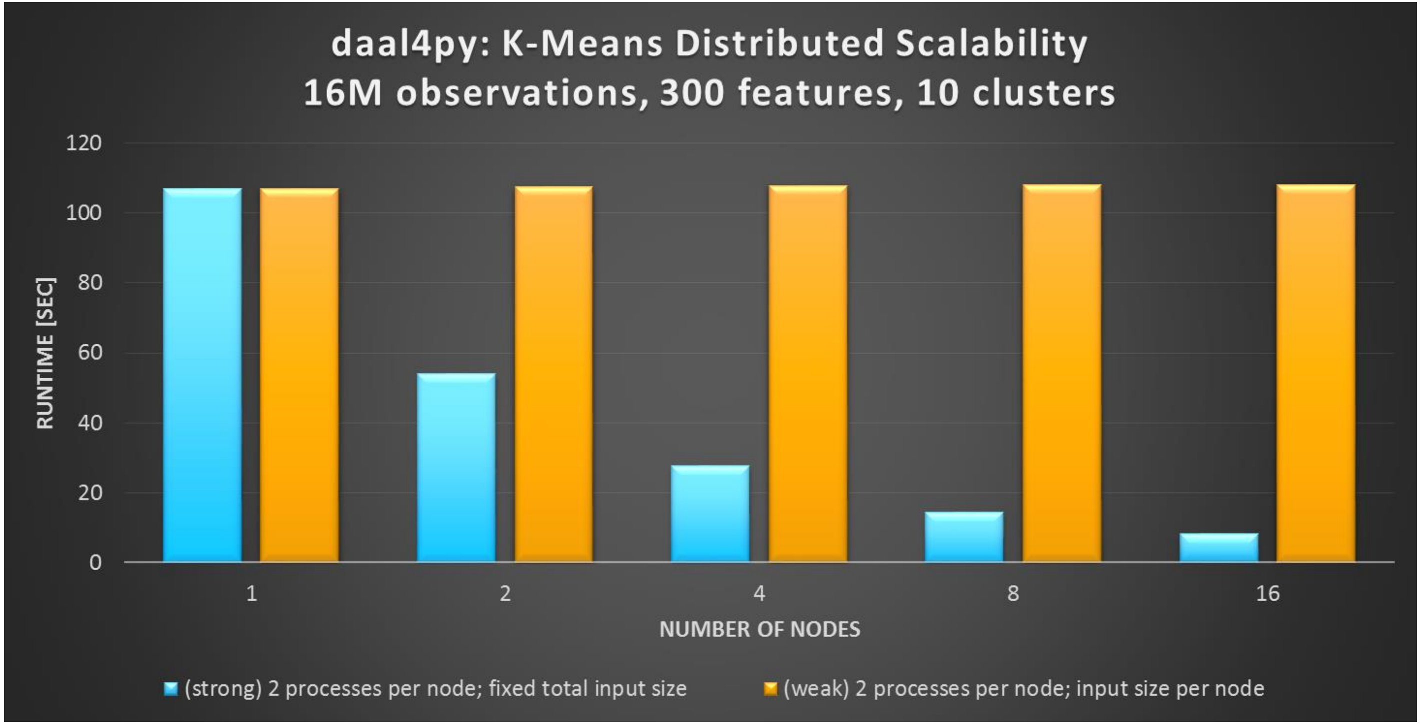
Try it out! `conda install -c intel/label/test daal4py`

Distributed K-Means using Daal4py

```
import daal4py as d4p

# initialize distributed execution environment
d4p.daalinit()

# load data from csv files into numpy arrays
files = ["kmeans_dense.csv", …]
dfin = [loadtxt(x, delimiter=',') for x in files]

# compute initial centroids & kmeans clustering
centroids = d4p.kmeans_init(10, t_method="plusPlusDense", distributed=True)
result = d4p.kmeans(10, distributed=True).compute(dfin, centroids.compute(dfin))
```

```
mpirun -n 4 -genv DIST_CNC=MPI python ./kmeans.py
```

Strong & Weak Scaling of K-Means via Daal4py



Hardware	Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz, EIST/Turbo on
	2 sockets, 20 Cores per socket
	192 GB RAM
	16 nodes connected with Infiniband
Operating System	Oracle Linux Server release 7.4
Data Type	double

Productivity with Performance via Intel® Python*

Intel® Distribution for Python*

scikit learn | NumPy | SciPy | Numba | pandas $y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$ | mpi4py | SMP | •••

Easy, out-of-the-box access to high performance Python

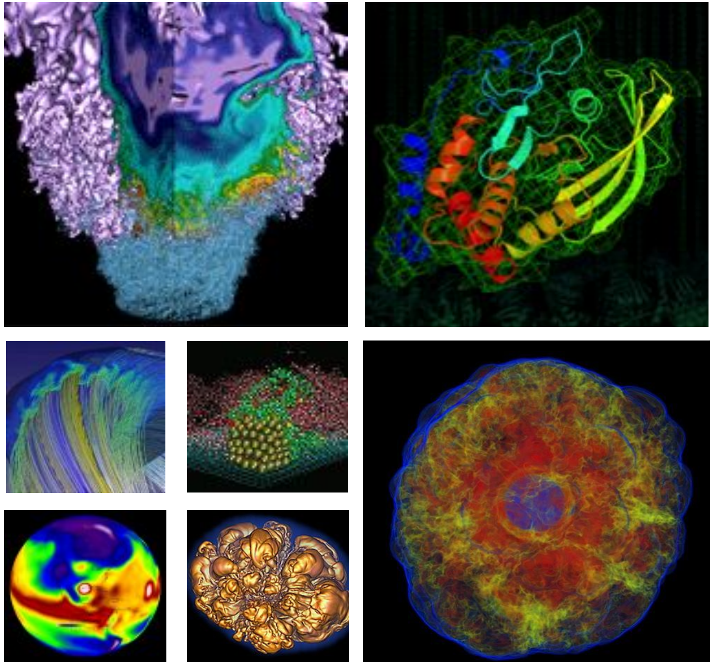
- Prebuilt accelerated solutions for data analytics, numerical computing, etc.
- Drop in replacement for your existing Python. No code changes required.

Learn More: software.intel.com/distribution-for-python

Software

Legal Disclaimer & Optimization Notice

Python at NERSC



Rollin Thomas

NERSC Data and Analytics Services

IXPUG

2018-05-10

NERSC

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Outline

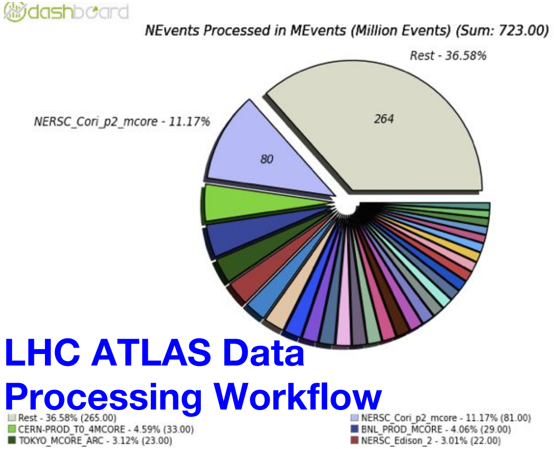
1. **Python enables HPC science at NERSC**
   Orchestration • Workflows • Analytics • HPC Apps

2. **How we help Python users at NERSC**
   Productivity • Performance

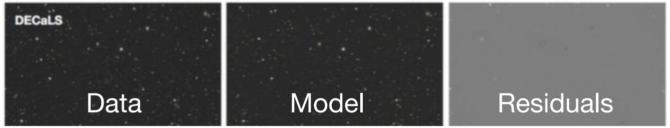
3. **Experimental/Observational Science Engagements**
   Python in NESAP for Data Projects w/Intel

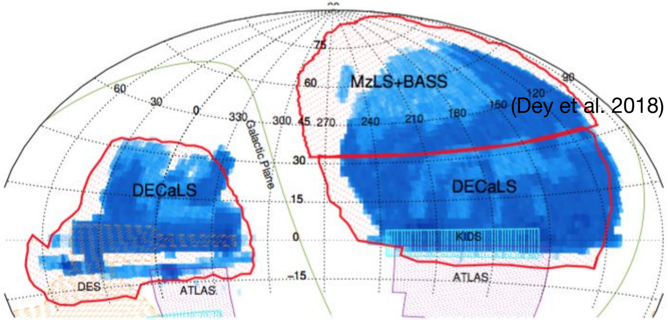
Science via Python@NERSC

Powering Workflows to Understand Properties of Materials



LHC ATLAS Data Processing Workflow

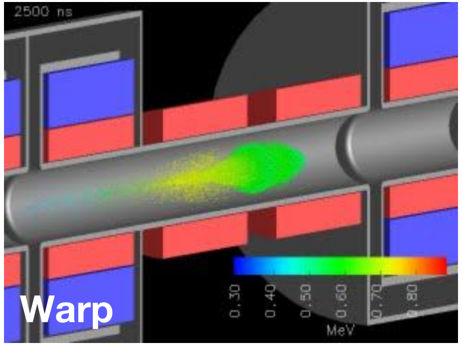




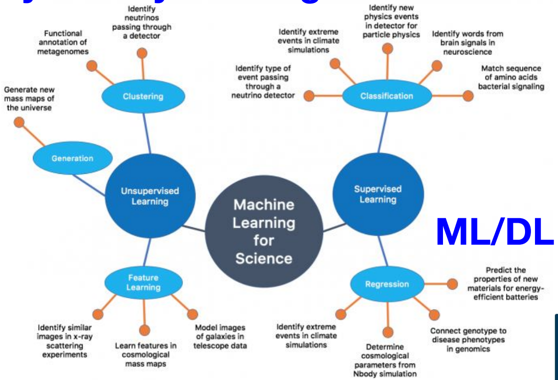
Sky Survey Catalogs for Cosmology

NBODYKIT

Modeling Dark Matter and Dark Energy



PIC Code for Plasmas and High Current Particle Beams



ML/DL

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Python in Edge Services

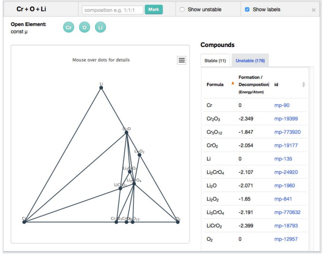
enables science through . . .

Data Sharing Across Facilities

Interfaces to HPC resources & workflows

MATERIALS PROJECT

FireWorks

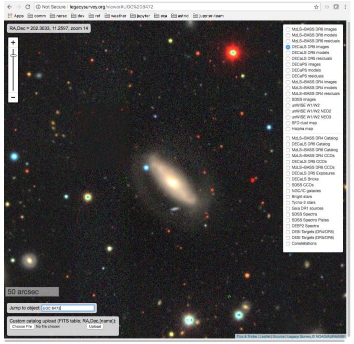


Interactive Tools

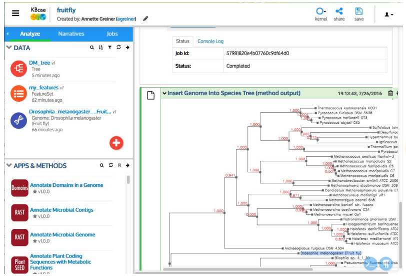
The Legacy Surveys

The Legacy Surveys are producing an inference model catalog of the sky from a set of optical and infrared imaging data, comprising 14,000 deg² of extragalactic sky visible from the northern hemisphere in three optical bands (g, r, z) and four infrared bands. The sky coverage is approximately bounded by $-18° < \delta < +84°$ in celestial coordinates and $|b| > 18°$ in Galactic coordinates. To achieve this goal, the Legacy Surveys are conducting 3 imaging projects on different telescopes, described in more depth at the following links:

The Beijing-Arizona Sky Survey (BASS)	*The DECam Legacy Survey* (DECaLS)	*The Mayall z-band Legacy Survey* (MzLS)



Rich Visualizations and UIs

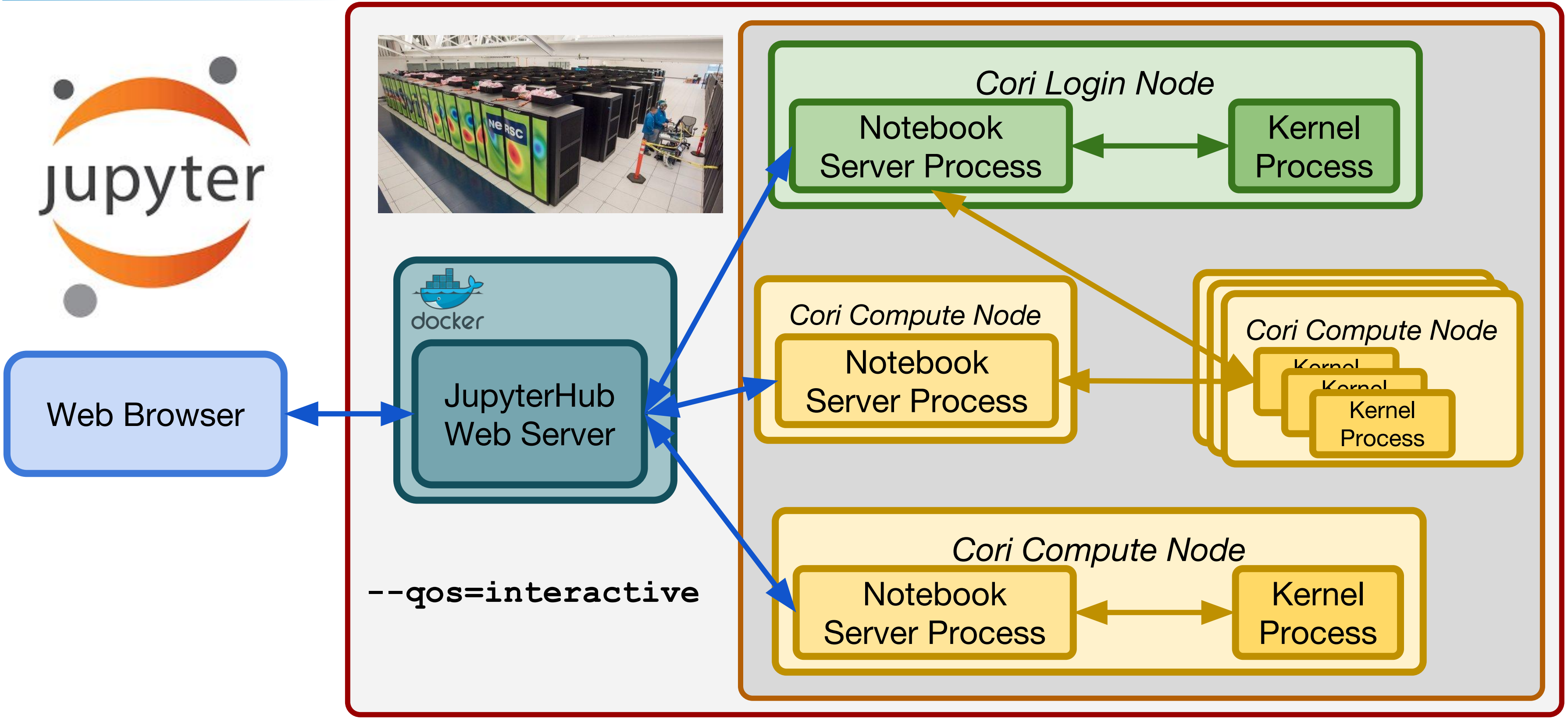


KBase PREDICTIVE BIOLOGY

U.S. DEPARTMENT OF ENERGY | Office of Science

BERKELEY LAB

Interactive Supercomputing



Python in HPC Jobs at NERSC

Around 3% of NERSC hours on Cori in the past year easily detected as Python jobs*:

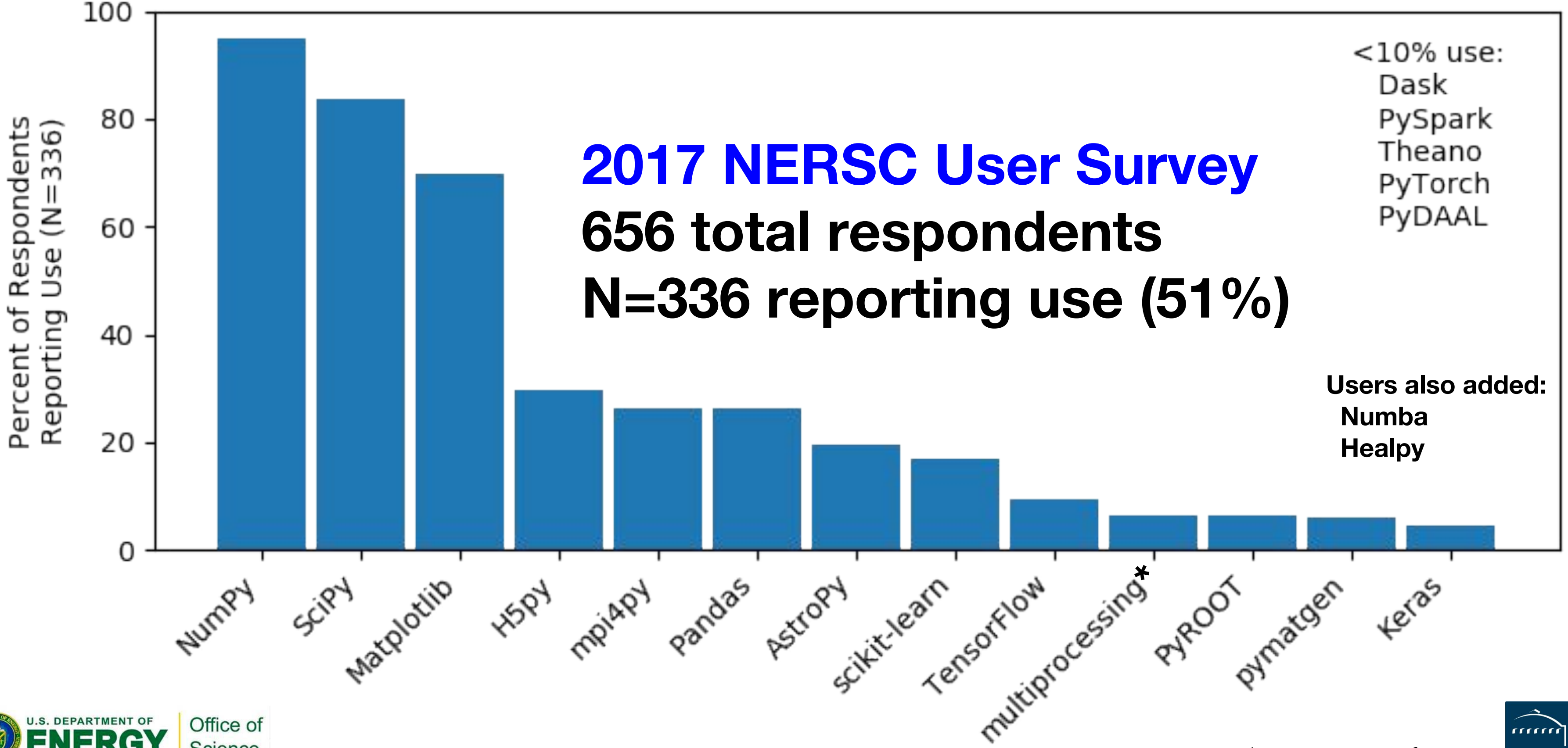
```
srun -n … python whatever.py …
```

***This is a lower limit*, as users:**

- **Often make main programs executable**
- **Use Python in containers to scale up**

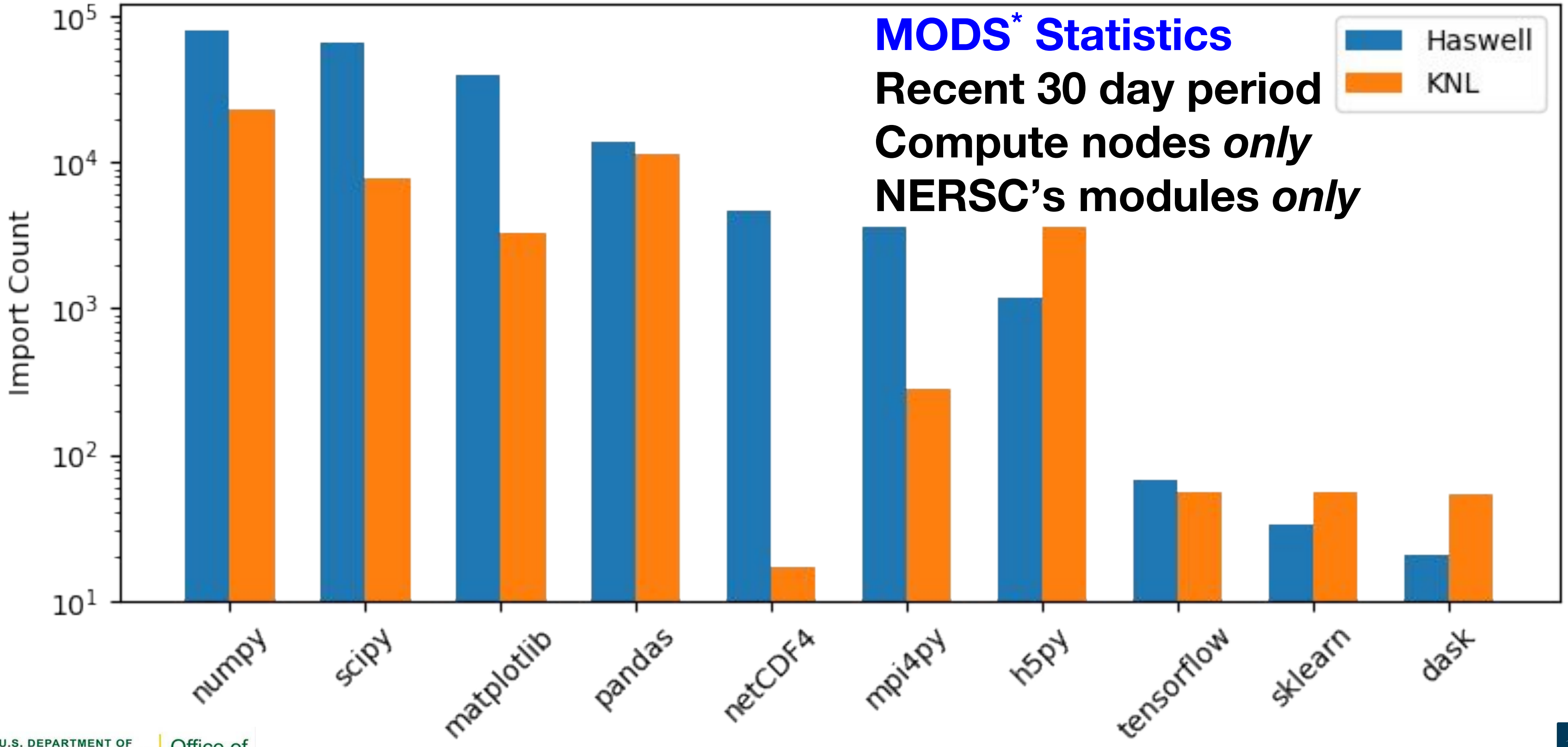
* Production batch *jobs*, not use on shared login nodes.

Packages Users Say They Use



* or concurrent.futures

Monitored Imports (Cori)



* MODS = Monitoring of Data Services at NERSC = BI Project in DAS

NERSC's Python Strategy

Focus on user productivity.
Support familiar, trusted, up-to-date libraries.
Find ways to put performance in user reach.

Examples:

Threaded libraries:	Intel MKL
Support cluster scaling:	Cray+mpi4py
Close architecture gaps:	Containers

NERSC Python: Anaconda

Most well-known and widely used distribution.
Designed around analytics, statistics, ML/DL.
“Personalized” environments and package manager.
Easily provide access to Intel Python Distribution.

2016: MKL added, and Intel upstreams optimizations:
NERSC drops its builds of Python on Cray the same year.

Other options for HPC:
Source builds, Spack, etc.

Handling MPI with mpi4py

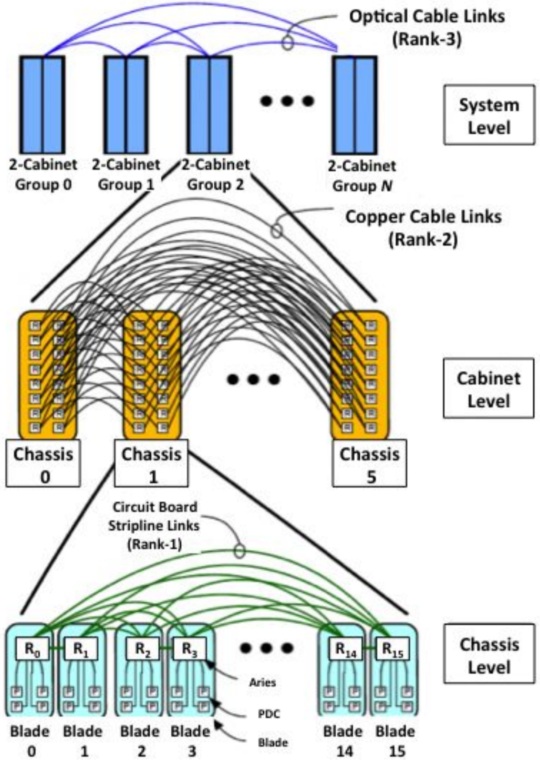
Cluster parallelism with MPI via mpi4py:

- MPI-1/2/3 specification support
- OO interface ~ MPI-2 C++ bindings
- Point-to-point and collectives
- Picklable Python objects & buffers

Build mpi4py & dependents with Cray MPICH:

```
python setup.py build --mpicc=cc
python setup.py install
```

Cray-provided Compiler wrapper



Cori Aries Interconnect

Containers

and Python go well together at NERSC

Motivations, esp. for data science:

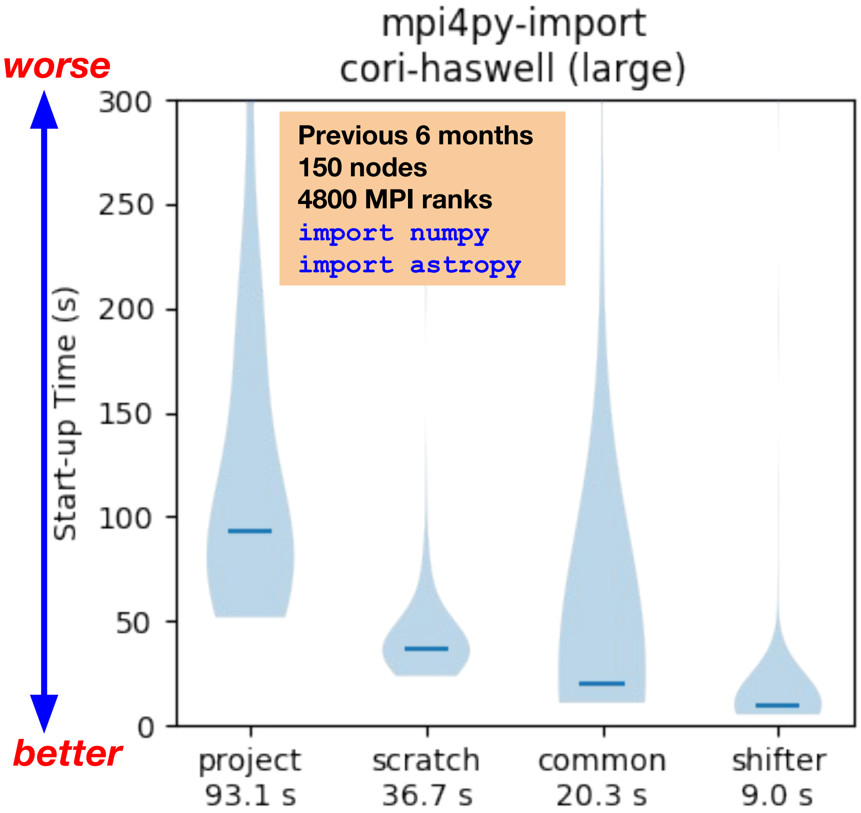
Flexibility Convenience
Consistency Reproducibility

Some Options:

Docker *Shifter (~Docker on Cray)*
Singularity CharlieCloud

Nice recent blog summary of the state of HPC containers:
https://www.stackhpc.com/the-state-of-hpc-containers.html

"Slow Launch" at Scale



[Median launch time incl. MPI_Init()]

Python's import is metadata intensive,
⇒ catastrophic contention at scale
⇒ it matters where you put your env

Project (GPFS):
For sharing large data files

Scratch (Lustre):
OK, but gets purged periodically!

Common (GPFS):
RO w/Cray DVS client-side caching
Open to users now, was only staff

Shifter (Docker Containers):
Metadata lookup only on compute
Storage on compute is RAM disk
ldconfig when you build image

Python on Knights Landing

Things will work, but at least,

- Understand and use numpy array syntax, broadcast rules, and scalar/“vector” interfaces to functions.
- Use threaded+vectorized libraries and compiled extensions, minimize time outside of using them.
- There may, in fact, be more than one way to do it; Prepare to rethink algorithms, memory usage, etc.
- Layer use of profiling tools to identify/assess hotspots.

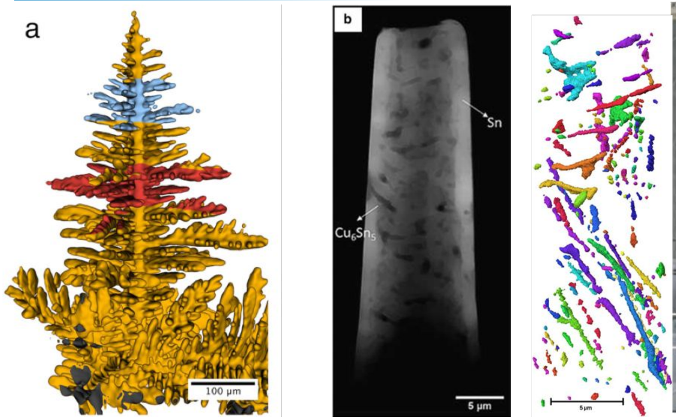
NESAP for Data

NERSC Exascale Science Applications Program for Data:

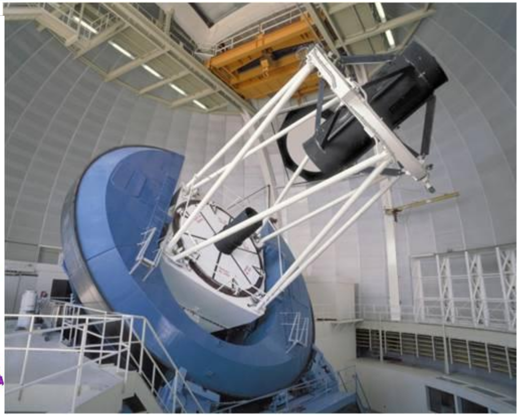
Users whose applications process, analyze, and/or simulate data sets or data streams from experiments and instrumentation supported by DOE need help preparing for extreme scale and exascale computing.

- Early Engagement with Code Teams
- Close Interactions with Vendors
- Developer Workshops, “Dungeons”
- Leverage Community Efforts
- Expanded Access to KNL + Data Ecosystem
- Postdoc Fellowship Program
- Training Docs, Online Modules

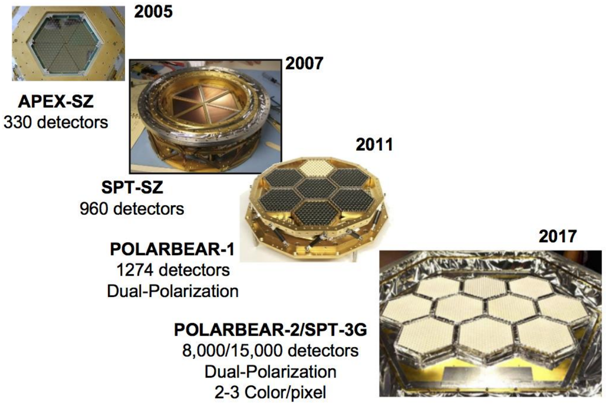
Python NESAP for Data Projects



TomoPy



DESI



TOAST

TomoPy (Python & C):

Tomographic data processing and image reconstruction

PI: Doga Gursoy, Argonne National Laboratory

DESI Pipeline (As Pure Python as Reasonably Possible):

Baryon acoustic oscillations (DESI Project)

PI: Stephen Bailey, Lawrence Berkeley Laboratory

TOAST (Time Ordered Astrophysics Scalable Tools, Python & C++):

Cosmic microwave background data analysis and simulation (CMB S4)

PI: Julian Borrill, Lawrence Berkeley Laboratory

DESI [Bailey, Stephey; Pavlyk, Douyeb, Fernandez, Hogan]

Science Purpose: Spectroscopy for Dark Energy science

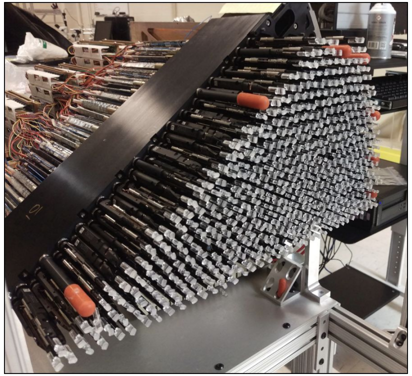
- 3D map of the Universe over 10 billion years
- Spectra of 10’s of millions of galaxies and quasars
- *Create flux-calibrated 1D tables of flux vs wavelength of Galaxies, quasars, etc. from 2D CCD image frames*

Algorithms and Methods

- Scientific Python stack (NumPy, SciPy, etc.; threaded)
- Linear algebra (esp. Hermitian eigen-decomposition)
- Special function evaluations, fitting functions to data
- MPI (mpi4py) data-parallel processing + Shifter to scale up

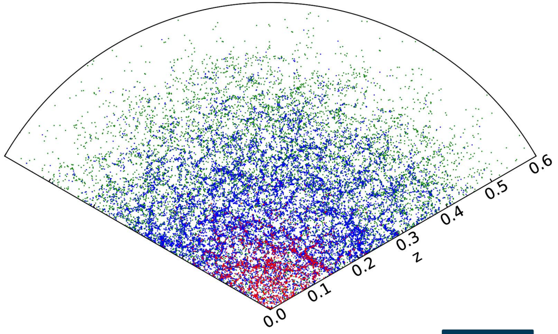
Production Requirements

- Real-time pressure to do real-time survey planning each day



DESI Fiber Positioner Petal

1 Exposure = 30 Frames = 15,000 Traces



DESI Optimization & Scaling

Simulation Code (Simulate Spectra on CCDs): 1.5-1.7x on HSW, multi-node scaling w/MPI

- Numba JIT compilation to speed up 2 lines of expensive matrix slicing
- MPI work to scale up the code:
  - Broadcast/reduce to scatter/gather where best use, complete initial I/O faster
  - Multi-level Comm scheme to optimally fill nodes
  - Scale tests up to 60 nodes so far, will be used in production soon
  - Single exposure (30 frames simultaneously) in 8 minutes
  - Roughly equal performance between multiprocessing and MPI on single node

Main Extraction Code (1D traces from CCD images)

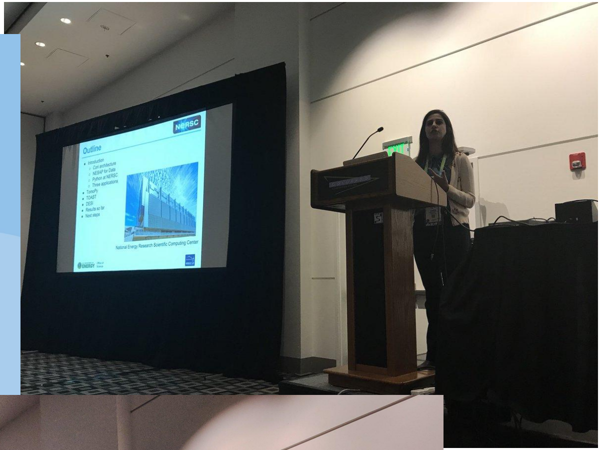
- Main bottleneck is `legval` in NumPy (scalar/vector args) observed at first Dungeon.
- ***Precompute `legval` w/large vector input (not scalar): promising but delicate refactor.***
- ***Also `legval` itself: 4x speedup with loop unrolling and Numba.***
- Using some of the code as a testbed for initial experimenting with PyPy.

PyHPC 2018

NERSC

At SC18!

SC18
Dallas, TX
hpc inspires.



8th Workshop on Python for High-Performance and Scientific Computing

Conclusion

NERSC

Python fills numerous critical roles at HPC scientific computing centers like NERSC.

Especially true in experimental/observational sciences, data processing/analysis more than analytics for now.

Achieving good Python performance is challenging and users (not often HPC-oriented) need to partner with center staff and vendors/developers to get it.